UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/754,785	01/04/2001	Pierre-Alain Darlet	11283/35	3238

30636	7590	05/31/2006

FAY KAPLUN & MARCIN, LLP
150 BROADWAY, SUITE 702
NEW YORK, NY 10038

EXAMINER
KISS, ERIC B

ART UNIT	PAPER NUMBER
2192	

DATE MAILED: 05/31/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Application Number: 09/754,785
Filing Date: January 04, 2001
Appellant(s): DARLET, PIERRE-ALAIN

MAILED

MAY 3 1 2006

Technology Center 2100

Michael J. Marcin
<u>For Appellant</u>

EXAMINER'S ANSWER

This is in response to the appeal brief filed March 3, 2006 appealing from the Office action

mailed August 25, 2005.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings

which will directly affect or be directly affected by or have a bearing on the Board's decision in

the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in

the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is

substantially correct. However, upon further consideration of the amendment filed December

27, 2005 and entered on appeal, (*see* Advisory Action (02/03/2006) at item 7,) one ground of

rejection has been withdrawn:

WITHDRAWN REJECTIONS

The following grounds of rejection are not presented for review on appeal because they

have been withdrawn by the examiner. The rejection of claims 36 and 37 under 35 U.S.C. § 101

is withdrawn. Claims 1-60 remain on appeal based on grounds II and III as stated in Appellant's

brief. (Appeal Brief at pp. 5-6.)

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Glen Overby, "Upgrading Your Minix System," 1990 [online], accessed 01/12/2006,

Retrieved from Internet <URL: http://www.funet.fi/pub/minix/unsorted/upgrading.txt>, 10 pages

(hereinafter "the Overby reference").

John Levine, "Linkers and Loaders, chapter 6," June 1999 [online] accessed 08/15/2005,

Retrieved from Internet <URL: http://www.iecc.com/linker/linker06.txt>, 9 pages (hereinafter

Levine or "the Levine reference").

6,185,733   BRESLAU et al.   2-2001

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1-41 and 43-60 are rejected under 35 U.S.C. 102(b) as being anticipated by the

Levine reference.

Claim 42 is rejected under 35 U.S.C. 103(a) as being unpatentable over the Levine

reference in view of U.S. Patent No. 6,185,733 to Breslau et al. (hereinafter *Breslau et al.* or "the

Breslau patent").

(10) Response to Argument

Appellant's Arguments Regarding the Rejection of Claims 36 and 37 under 35 U.S.C. § 101 Are Moot in View of the Withdrawal of This Rejection.

The rejection of claims 36 and 37 under 35 U.S.C. § 101 has been withdrawn. (*See supra*

item 6.) Accordingly, Appellant's arguments for the reversal of this rejection are now moot.

(Appeal Brief at pp. 6-7.)

The Rejection of Claims 1-41 and 43-60 under 35 U.S.C. § 102(b) Should Be Affirmed Because the Levine Reference Anticipates These Claims.

Claim 1

Appellant has stated, "Claims 1-60 may stand or fall together." (Appeal Brief at p. 6.)

Accordingly, claim 1 may be considered as a representative claim, and for the reasons that

follow, claim 1 is anticipated by the Levine reference.

Claim 1 is recited as,

A method, comprising:
   receiving a software module, the software module including references to
location within the software module, at least some of the references being
backward references; and
   reordering components of the software module to remove at least some of
the backward references.

Appellant first argues that the Levine reference fails to teach reordering components of a

software module to remove backward references because the Levine reference purportedly, "fails

to address the rearrangement of headers, sections, tables, and various other components to

convert of a software module for efficient linking and loading of the software module . . . ."

(Appeal Brief at pp. 8-9.)

In response to Appellant's argument that the Levine reference fails to show certain

features of Appellant's invention, it is noted that the features upon which Appellant relies (i.e.,

the rearrangement of headers, sections, tables, and various other components to convert of a

software module for efficient linking and loading of the software module) are not recited in

rejected claim 1. (*Id.*) Although the claims are interpreted in light of the specification,

limitations from the specification are not read into the claims. *See In re Van Geuns*, 988

F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993). Further, the examiner notes that where an explicit

definition is provided by the applicant for a term, that definition will control interpretation of the

term as it is used in the claim. *Toro Co. v. White Consolidated Industries Inc.*, 199 F.3d 1295,

1301, 53 USPQ2d 1065, 1069 (Fed. Cir. 1999) (meaning of words used in a claim is not

construed in a "lexicographic vacuum, but in the context of the specification and drawings").

However, in this case, Appellant's specification does not set forth a definition of *components*

"with reasonable clarity, deliberateness, and precision" that would render the incorporation of

such a definition into the claims appropriate. *See In re Paulsen*, 30 F.3d 1475, 1480, 31 USPQ2d

1671, 1674 (Fed. Cir. 1994). Accordingly, the limitations of claim 1 cannot be properly

construed to require "the rearrangement of headers, sections, tables, and various other

components to convert of a software module for efficient linking and loading of the software

module," as Appellant apparently contends. (*See* Appeal Brief at pp. 8-10).

Moreover, despite Appellant's argument that the *lorder* and *tsort* functions described in

the Levine reference, "provide nothing more than a means of ordering object files on the basis of

symbol dependencies during the creation of an archive library," (Appeal Brief at p. 9,) the

examiner maintains that this functionality anticipates claim 1. As disclosed in the Levine

reference,

> *Lorder* **took as its input a set of object files** (not libraries), and produced a
> dependency list of what files refer[r]ed to symbols in what other files. . . . *Tsort*

> did a topological sort on the output of *lorder*, **producing a sorted list of files so each symbol is defined after all the references to it,** allowing a single sequential pass over the files to resolve all undefined references. The **output of *lorder* was used to control *ar*.**

Levine at p. 5 (emphasis added). Thus, the Levine reference discloses: (1) "receiving a software module," *i.e.*, taking <u>a set of object files</u> as input; (2) "the software module including references to locations within the software module," *i.e.*, the dependency list is based on what object files refer to symbols in other object files; (3) "at least some of the references being backward references" and "reordering components of the software module to remove at least some of the backward references," *i.e.*, the purpose of the topological sort is to sort the input files such that there are no backward references (symbols defined before references to them), and this sorted list is processed by *ar* to store the reordered object code. (*See Id.*; *see also* Claim 1.)

To further emphasize that this interpretation of the Levine reference is appropriate, the examiner previously provided an additional reference, the Overby reference, (*see* Advisory Action (02/03/2006) at item 13,) which also describes the use of *lorder* and *tsort* in an equivalent manner to that disclosed by the Levine reference. Specifically, the Overby reference discloses,

> Two utilities are required to create a library order: lorder and tsort. Lorder creates a dependency list, that is, a list of what functions are required by what other functions. Tsort takes the output of lorder and does a "topological sort" **to create an ordering with no backwards references.**

Overby at p. 4 (emphasis added). Although not applied in the rejection of the claims, this reference illustrates that the elimination of backwards references is indeed the functionality of the Levine reference disclosure.

The examiner submits that the Levine reference, as discussed above, anticipates claim 1, and that the rejection of claim 1 under 35 U.S.C. § 102(b) should be affirmed. Accordingly, the

examiner submits that the rejection of claims 2-41 and 43-60, grouped together with claim 1 by

Appellant (Appeal Brief at p. 6,) should likewise fall with claim 1. However, it is also noted that

Appellant has provided further arguments with respect to claims 9, 16, 23, 36, 38, 39, and 55.

These arguments are addressed below.

Claim 9

Appellant has merely restated the limitations recited in dependent claim 9 and further

asserts, "[F]or at least the reasons discussed above with respect to claim 1, claim 9 should also be

allowed." Appellant has not alleged that claim 9 should be considered patentable for any reason

separate from the reasons given for claim 1. A statement which merely points out what a claim

recites will not be considered an argument for separate patentability of the claim. 37 CFR

41.37(c)(vii). Accordingly, the rejection of claim 9, and claims 10-15 dependent therefrom,

should be affirmed for the reasons given above with respect to claim 1.

Claim 16

Appellant alleges that the Levine reference, "fails to teach or disclose the method of

linking the software module onto a target memory space or using symbol resolution without

storing the entire software module in local memory." (Appeal Brief at p. 11.) However, it

should be noted that the Levine reference discusses libraries (like the one created using the ar

function) in the context of linkers and loaders, *i.e.*, utilities to link the software module onto a

target memory space via symbol resolution. *See, e.g., Levine* at p. 1 ("Every modern linker

handles libraries, collections of object files that are included as needed in a linked program. . . .

Once loaders and linkers started to resolve symbolic references, it became possible to automate

the process by selecting routines from the library that resolve otherwise undefined symbols.").

Further, the lorder and tsort functions allow the resulting library to be organized in such as way

as to allow a single sequential pass over the files to resolve all undefined references. *Levine* at p.

5. During this sequential pass, the linker will include the appropriate object files as appropriate.

Id. at p. 6. Thus, there is no need to store the entire module in memory during this single

sequential pass because the linker only has to search forward (sequentially) through the library.

Accordingly, the examiner submits that the Levine reference anticipates claim 16, and this

rejection should be affirmed.

Claim 23

Appellant has incorporated the arguments with respect to claim 16 into the argument for

claim 23. Accordingly, the rejection of claim 23, and claims 24-35 dependent therefrom should

be affirmed for the reasons given above with respect to claim 16.

Claim 36

Appellant has incorporated the arguments with respect to claim 16 into the argument for

claim 36. Accordingly, the rejection of claim 36, and claim 37 dependent therefrom should be

affirmed for the reasons given above with respect to claim 16.

Claim 38

Appellant has merely restated the limitations recited in claim 38 and further asserts,

"[F]or at least the reasons discussed above with respect to claim 1, claim 38 should also be

allowed." Appellant has not alleged that claim 38 should be considered patentable for any

reason separate from the reasons given for claim 1. A statement which merely points out what a

claim recites will not be considered an argument for separate patentability of the claim. 37 CFR

41.37(c)(vii). Accordingly, the rejection of claim 38 should be affirmed for the reasons given

above with respect to claim 1.

Claim 39

Appellant has incorporated the arguments with respect to claim 16 into the argument for

claim 39. Accordingly, the rejection of claim 39 should be affirmed for the reasons given above

with respect to claim 16.

Claim 55

Appellant has incorporated the arguments with respect to claim 1 into the argument for

claim 55. Accordingly, the rejection of claim 55, and claims 56-60 dependent therefrom should

be affirmed for the reasons given above with respect to claim 1.

The Rejection of Claim 42 under 35 U.S.C. § 103(a) Should Be Affirmed Because the Primary Teaching of the Levine Reference Anticipates Claim 1, and Appellant Has Not Alleged Any Error in the Secondary Teachings of Breslau as Applied to Claim 42.

The thrust of Appellant's argument for claim 42 is that the Breslau patent allegedly fails

to cure the defects of the Levine reference with respect to the limitations recited in parent claim

1. Appellant does not argue any error in the applied teachings of the Breslau patent as providing,

"transferring the reordered module to a different computer system and linking the module on the

different computer system." (*See* Appeal Brief at pp. 15-16.) Accordingly, the rejection of

claim 42 should be affirmed for the reasons given above with respect to claim 1.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related

Appeals and Interferences section of this examiner's answer.

(12) Text of the Final Rejection

Claims 1-41 and 43-60 are rejected under 35 U.S.C. 102(b) as being
anticipated by John Levine, "Linkers and Loaders, chapter 6," June 1999
[online] accessed 08/15/2005, Retrieved from Internet <URL:
http://www.iecc.com/linker/linker06.txt>, 9 pages (hereinafter *Levine*).

As per claim 1, *Levine* discloses receiving a software module, the software
module including references to locations within the software module, at least
some of the references being backward references; and reordering components of
the software module to remove at least some of the backward references (see
"Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and
lorder to arrange object files within an archive library in proper dependency order
to allow a single sequential linker pass to resolve all undefined references).

As per claim 2, *Levine* further discloses adjusting at least one of the
references in the software module to reflect the reordering of the components of
the software module, so that the at least one of the references remains a reference
to the same component, by to the component's new, reordered location, the new,
reordered location coming after the at least one reference in the software module
(see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 3 and 4, *Levine* further discloses the software module
including a symbol table, the symbol table including no backward references after
the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in
particular, the discussion of using tsort and lorder to arrange object files within an
archive library in proper dependency order to allow a single sequential linker pass
to resolve all undefined references). Further, *Levine* discloses that under certain
circumstances, lorder and tsort won't be able to come up with a total order for the
files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 5-8, *Levine* further discloses the use of relocatable ELF
object files, which include sections grouped into segments (see "Library formats"
on pp. 1-5). *Levine* further discloses the software module including a symbol
table, the symbol table including no backward references after the reordering and
adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the
discussion of using tsort and lorder to arrange object files within an archive
library in proper dependency order to allow a single sequential linker pass to
resolve all undefined references). Further, *Levine* discloses that under certain
circumstances, lorder and tsort won't be able to come up with a total order for the
files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claim 9, *Levine* discloses a reorder module configured to receive a
software module including references to locations within the software module, at
least some of the references being backward references, the reorder module
configured to reorder components of the software module and remove at least
some of the backward references (see "Creating libraries" on pp. 5-6, and in
particular, the discussion of using tsort and lorder to arrange object files within an

archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 10, *Levine* further discloses adjusting at least one of the references in the software module to reflect the reordering of the components of the software module, so that the at least one of the references remains a reference to the same component, by to the component's new, reordered location, the new, reordered location coming after the at least one reference in the software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 11 and 12, *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claims 13-15, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5). *Levine* further discloses the software module including a symbol table, the symbol table including no backward references after the reordering and adjusting steps (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, *Levine* discloses that under certain circumstances, lorder and tsort won't be able to come up with a total order for the files, resulting in backward references remaining (see "Exercises" on p. 8).

As per claim 16, *Levine* discloses receiving a software module sequentially, the software module having at least one symbol reference; linking the software module onto a target memory space; and resolving the at least one symbol reference without storing the entire software module in local memory while the symbol reference is resolved (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 17-22, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 23, *Levine* discloses a linker configured to sequentially receive a software module having at least one symbol reference, the linker configured to resolve the symbol reference, the linker configured to store less than the entire software module in local memory during the resolution of the at least one symbol reference (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive

library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 24-27, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5; see further, "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 28 and 29, *Levine* further discloses a system symbol table including a field indicative of a defining software module (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claims 30 and 31, *Levine* further discloses a software module list (see "Library formats" on pp. 1-5).

As per claims 32-34, *Levine* further discloses storing link status in a symbol table (see "Library formats" on pp. 1-5).

As per claim 35, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 36, *Levine* discloses a symbol table, the symbol table including at least one backward reference; at least one component of the software module located before the symbol table in the software module; wherein none of the at least on components of the software module located before the symbol table include backward internal references (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

As per claim 37, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claim 38, *Levine* disclose receiving a software module, the software module including references to locations within the software module, at least some of the references being backward references; and reordering the components of the software module to remove at least some of the backward references (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

As per claim 39, *Levine* discloses receiving a software module sequentially, the software module having at least one symbol reference; linking the software module onto a target memory space; and resolving the at least one

symbol reference without storing the entire software module in local memory at one time (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references). Further, the use of a computer-readable medium (such as memory) is inherent (and necessary) in realizing the computer-implemented functionality described in *Levine*.

As per claims 40 and 41, *Levine* further discloses linking the reordered module after the reordering (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 43-46, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

As per claims 47-54, *Levine* further discloses the reference pointing to/into a section or module before and after reordering (see "Creating libraries" on pp. 5-6 and "Library formats" on pp. 1-5).

As per claim 55, *Levine* discloses receiving a software module, the software module including components arranged in a first order, a first one of the components including a reference to a location in a second one of the components, the second one of the components preceding the first one of the components in the first order; and arranging the components into a second order so that the second one of the components is subsequent to the first one of the components in the second order (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claims 56 and 57, *Levine* further discloses linking the reordered module after the reordering (see "Creating libraries" on pp. 5-6, and in particular, the discussion of using tsort and lorder to arrange object files within an archive library in proper dependency order to allow a single sequential linker pass to resolve all undefined references).

As per claim 58-60, *Levine* further discloses the use of relocatable ELF object files, which include sections grouped into segments (see "Library formats" on pp. 1-5).

Claim 42 is rejected under 35 U.S.C. 103(a) as being unpatentable over *Levine*, as applied to claim 1 above, and further in view of U.S. Patent No. 6,185,733 to Breslau et al.

As per claim 42, *Levine* discloses such a method but fails to expressly disclose transferring the reordered module to a different computer system and linking the module on the different computer system. However, *Breslau et al.* teaches the use of remote object libraries distributed prior to linking (see, for

example, col. 4, lines 11-20). Therefore, it would have been obvious to one of ordinary skill in the computer art at the time the invention was made to such use of a different computer for linking. One would be motivated to do so, for example, to facilitate distributed software development efforts or reduce the physical storage requirements for object files (see, for example, col. 2, lines 4-25).
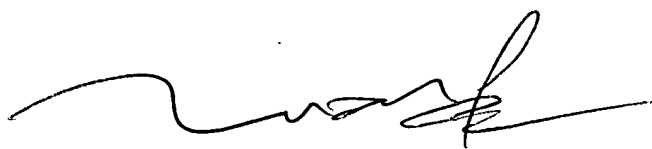
For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Eric B. Kiss

Conferees:

Tuan Q. Dam

TUAN DAM
SUPERVISORY PATENT EXAMINER

Wei Y. Zhen

WEI ZHEN
SUPERVISORY PATENT EX